John Raven

Project 2 Bundle

The first thing that I did for this project bundle was update my stats library with the new distributions that we learned. Then I created a poker hand tester package, that tests the different probabilities of getting different poker hands. Next, I created the plotter, salter, and smoother classes, each of these reads a CSV file and outputs a new file with updated values to be graphed.

The first distribution was the binomial distribution (binDistro), this method accepts three variables that are adjusted by the user in the tester class. This distribution would be used if you have only two outcomes for the problem you are working with. For example, if you are tossing a coin, testing if something is a pass or fail, or trying to find the number of defective parts in a production warehouse.

The next distribution is the geometric distribution (geoDistro), this method accepts two variables that are adjusted by the user. The geometric distribution tests for the number of failures before the first success. For example, how many times you need to open a mystery box to get a specific prize, or how many times you need to flip a coin to get tails.

The next distribution is the hyper geometric distribution (hypGeoDistro), this method accepts 4 variables that are adjusted by the user. The hyper geometric distribution is used when selecting **n** items from a population or total without replacing the item selected. This can be used for finding the probability of drawing specific cards from a deck or selecting a specified amount of male and female workers.

The next distribution is the Poissan distribution (poissan), this method accepts two variables that are adjusted by the user. The Poissan distribution is used for modelling the number of events that occur in a specified time. For example, the number of times a computer may crash in a specific time span, or the number of accidents that occur in an intersection. This is an ideal distribution when the number of trials is much larger than the number of successes.

The last distribution is Chebyshev’s theorem (cheby), this method accepts three variables that are adjusted by the user. Chebyshev’s theorem is usually used when analyzing a data set and you want to find a percentage of values that fall between an upper and lower bound using the mean and standard deviation.

The next set of programs that were written were to test the probability of getting different poker hands in a hand of 5 cards, each of these tests were ran 10,000 times. The hands we were testing for were one pair, two pairs, three of a kind, four of a kind, straight, flush, and full house. There was a total of four different classes required for this package, a Card class, Deck class, HandEvaluator class, and a Poker\_Tester class. The Card class creates the Card objects that the Deck and HandEvaluator class use. The Card objects get assigned a number 1-13 and a suit, Clubs, Diamonds, Hearts, and Spades by the Deck class. The Deck class can also shuffle the deck it creates, print the organized deck, and draw a card from the deck. The HandEvaluator class is where all the calculations are done. This class contains a total of eight methods; result, pairTester, threeOfKind, twoPair, straightTester, fullHouseTester, flushTester, and fourOfKind. Each of these methods returns the probability of their respective hands in a five-card hand. The Poker\_Tester class sets the number of times each test should be ran.

The next program that needed to be written was the Plotter. The Plotter program needed to take in X values and plug it into a function to get the Y values. Once the Y values were found it needed to print both the X and Y values into a CSV file. Once the CSV file was created, we were required to manually create a graph using the values the program had printed out. The function I chose was y=mx+b due to how simple the function is and how easy it would be to spot errors.

The next program that was required was the Salter. The Salter program needed to read a file of X and Y values, then add or subtract a bound set by the user from the Y value and update the CSV file. Once updated the CSV file needed to be opened manually and a graph needed to be created using these values. I chose to use a line graph because I felt it showed the data the best even though the values were randomized. My program reads the Plotter.csv file created by the plotter and then “salts” the data by randomly adding or subtracting the bound set by the user. For my testing I chose 10 for my bound.

The next program that was required was the Smoother. The Smoother program needed to read a CSV file of X and Y values, then take in a range set by the user and use that range to average the Y values around the current Y value and replace the current Y value with the average. Once all the values had been replaced the CSV file needed to be opened and a graph needed to be manually created. Unfortunately, I was unable to complete this portion of the project bundle as I was unsure of how to go about it. Conceptually, I understand that while it’s looping through the CSV file it needs to look at the values before and after the current value, add them up then divide them by the range. However, I was unable to figure out how to convey this in the program.